Status Report on Spatial Data Structures

Brian Bush and Rob Oakes

Los Alamos National Laboratory

8 December 1994

PURPOSE

The purpose of this presentation is to make the TRANSIMS team aware of the ongoing work on spatial data structures. We also want to gather feedback from the team concerning their requirements for these data structures.

OUTLINE

- definitions
- use cases
- taxonomy
- design
- database size

THE PROBLEM OF DEFINITION

- In general, spatial data structures are data structures used to optimize the storage and querying of spatial data.

- The difficulty in defining a spatial data structure precisely lies in deciding which questions are “spatial” (the *functional* aspect of the definition) and which data is “spatial” (the *structural* aspect of the definition).

- Therefore, we will start with a minimal definition and then discuss various ways in which the definition can be expanded.

MINIMAL DEFINITION

- *Definition*
  - Spatial data structures are indices to data with spatial coordinates. Entries in the spatial data structure can be inserted, deleted, moved, and searched based on spatial coordinates.
  - Efficient containment, intersection, and nearness queries are directly supported.

- *Discussion*
  - Thus an intersection or road segment would be in a spatial data structure used by simulation pre- and post-processors. Vehicles would be in a (spatial or non-spatial) data structure, but a traveler probably would be in a non-spatial data structure. Furthermore, a data structure used to optimize vehicle to vehicle nearness will be crucial to the microsimulation while a data structure to optimize general roadway queries could be an unnecessary burden to the microsimulation.

  - To further distinguish between the two data structures mentioned, roadway data is by convention expressed in geographic coordinates, while vehicle data must be expressed in coordinates more useful to vehicle nearness queries.

ADDITION OF “ADJACENCY” TO THE DEFINITION

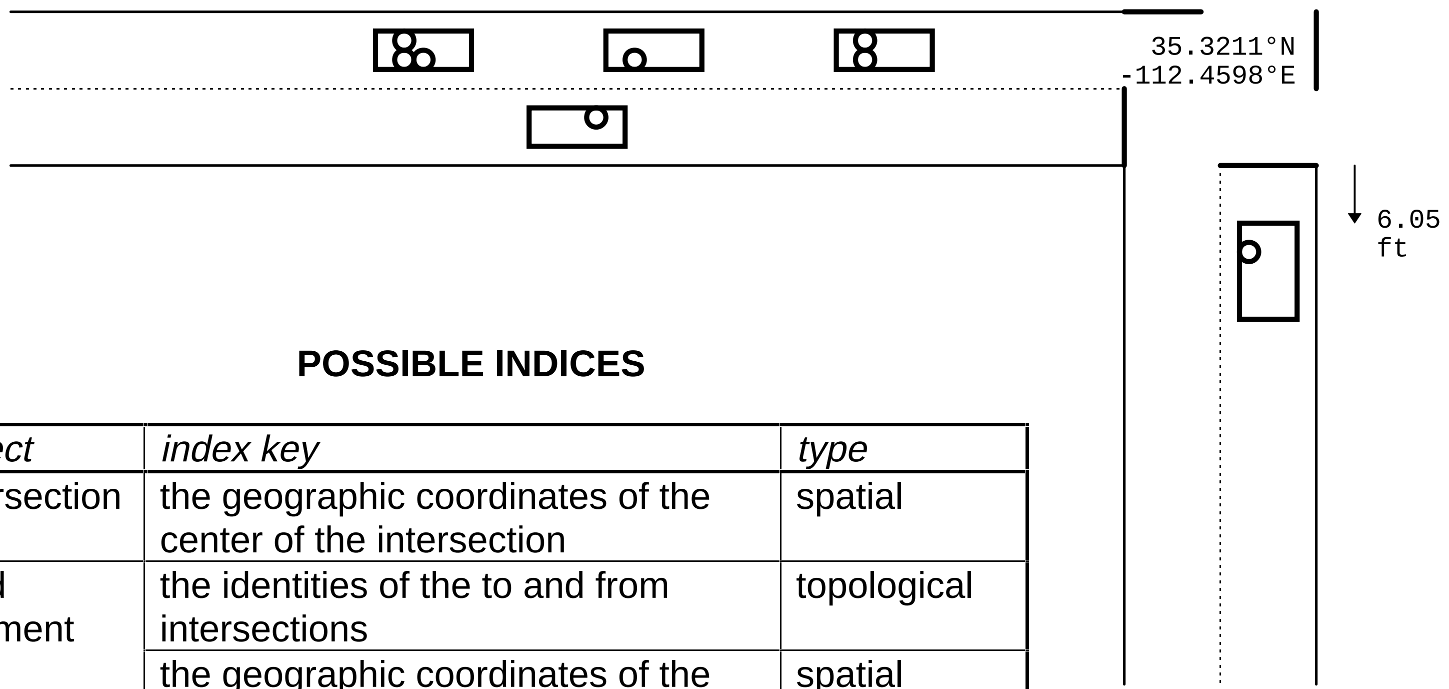
- By *adjacency* we mean that the spatial data structure stores what intersects/contains/is contained in/is near to what.

- This generalizes the containment, intersection, and nearness apsect of the minimal definition above.

- A spatial data structure would, for example, cross-reference all of the polygons in the structure with all of the points and lines, so that one could immediately determine which streets are located in a particular census block or in which land use area a given intersection lies, and vice versa.

ADDITION OF “TOPOLOGY” TO THE DEFINITION

- By *topology* we mean that the spatial data structure store what connects to what.
- Each intersection would be cross-referenced to the streets leading up to it and each onramp would be cross-referenced to the freeway it leads to, for instance.
- The question here is not whether of not network connections are needed—they certainly are—the question is whether they should be an intrinsic part of the spatial data structure or should be represented within another data structure that is more network-oriented.



POSSIBLE INDICES

object	*index key*	*type*
intersection	the geographic coordinates of the center of the intersection	spatial
road segment	the identities of the to and from intersections	topological
	the geographic coordinates of the endpoints	spatial
vehicle	the identity of road segment	nonspatial
	the identity of road segment and the offset along road segment	adjacency
	the geographic coordinates of the center of the vehicle	spatial
traveler	the identity of vehicle	nonspatial
	the geographic coordinates of the traveler	

ADDITION OF “CONTAINERSHIP” TO THE DEFINITION

- Another way to expand the definition of a spatial data structure is to make them *containers* rather than just indices.
- By *container* we mean that the spatial data structure has ownership of the data and is responsible for the persistence of the data.
- For example, when a street is added to an index, the index stores the street identification number in the spatial data structure, but when a street is added to a container, the container copies the whole street object into the spatial data structure (and presumably stores it on disk).
- Spatial containers provide for faster spatial searching and retrieval of data, but much slower non-spatial searching and retrieval.

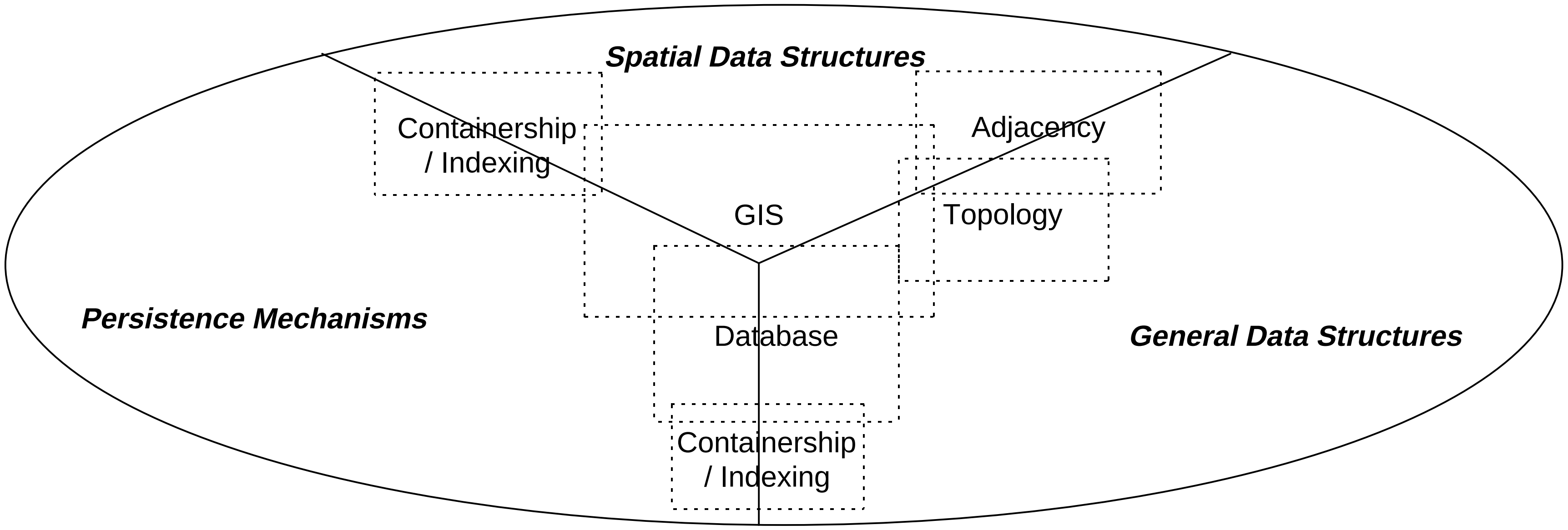
DEGREE OF UNIFORMITY IN SPATIAL DATA STRUCTURES

- Another issue related to defining a spatial data structure is specifying the degree of uniformity of the items in the structure.

- Should streets, census tracts, intersections, etc. all be in one structure, or should the network objects be separated from the demographic ones, or should every type of object have a separate data structure?

INTEGRATION WITH THE REST OF TRANSIMS

- At some point spatial data structures will have to be integrated into the persistence mechanism used to store data on disk.
- Spatial data structures, non-spatial data structures, and the persistence mechanism will have to fit together well into an overall framework for data handling in TRANSIMS.



USE CASES / SCENARIOS FOR SPATIAL DATA STRUCTURES

- a use case analysis is the process of[*]
  - establishing system boundaries
  - defining the mechanisms of the system
- examples
  - extracting part of a large road network for an simulation/analysis
  - converting spatial data provided by an MPO
  - building network topology from geographic data
  - disaggregation of attributes based on polygon overlap
  - estimating the visibility along road segments
  - subdividing a road network for initial computational distribution
  - displaying microsimulation output
  - selecting a geographic object on display
  - animating vehicle movement
  - producing a fundamental diagram
  - comparing traffic flows at different times

[*] After *Catalyst Solutions*.

Subdividing a Road Network for Initial Computational Distribution

Actor: Initial Load Balancer

Basic Course: The microsimulator is being initialized for a simulation that will be distributed over a number of computational processors (CPUs.) The road network, consisting of a number of roads-segments (arcs) and segment connectors (nodes) will be split into as many regions as there are CPUs. Since the density of the nodes and arcs is not homogeneous, measures will be taken to allow for differently sized regions. For example, the number of nodes assigned to each CPU can be in proportion to the anticipated performance of that CPU. A kd-subdivision of the road network will be used.

Alternatives: Arc count and arc length may also be used as predictors of computational demand for a given region. Alternative geometric or topological network subdivision may be considered.

Producing a Fundamental Diagram

Actor: Analyst

Basic Course: A fundamental diagram presents traffic flow vs. density over a path—one that has no entry or exit nodes other that the beginning and end nodes.

The analyst identifies a set of simulations for which to produce fundamental diagrams. In addition the analyst specifies (by graphical interaction with a map of the simulated road network) the path on which to compute the diagram and the time step to use. Then the toolbox computes and plots vehicle flow vs. density at specified times.

Alternatives: The analyst can request printed output of the images currently seen. Alternate methods for selecting a road segment are possible.

Converting Spatial Data Provided by an MPO

Actor: Analyst's Toolbox

Basic Course: The Analyst's Toolbox is converting data that a user has provided in an external format (e.g., ArcInfo or TIGER/Line format) into the internal format used by TRANSIMS. As the input files are parsed, geographic objects are created and added to persistent storage and indexed spatially (and otherwise). When the importation process is completed there will be one or more spatial indices to all of the network objects with geographic coordinates.

Alternatives: In the process of "building" geographic objects, it may be necessary to determine the connectivity of one geographic object with another; this will involve intersection, nearness, and containment queries. Also, it may be necessary to optimize the persistent storage for rapid access to geographic objects via spatial queries.

Disaggregation of Attributes Based on Polygon Overlap

Actor: Disaggregator

Basic Course: The Disaggregator needs to distribute data at one level of geographic detail to another level of geographic detail. An example of this would be taking employment statistics for a standard metropolitan statistical area (SMSA) and distributing it among the census tracts within the SMSA. The demographic or economic data associated with each polygon in the source database is disaggregated among the destination database's polygons that overlap it proportionally to the fraction of overlapping area. After the data is disaggregated, all the data is represented at the lower level of geographic detail and the original data can be recovered by reaggregating the data (i.e., reversing the disaggregation process).

Alternatives: The model for disaggregating the data may be more sophisticated than using area proportionality—it may involve additional weighting factors, points of attraction, an auxiliary set of polygons for weighting, etc.

SPATIAL DATA STRUCTURE TAXONOMY*

- Point Structures
  - Point Quad-Tree/Trie
  - Region Quad-Tree/Trie
  - 2d-Tree
- Extended Structures
  - Corner Stitching
  - Grid File
  - EXCELL Method
  - PLOP-Hashing
  - CIF Quad-Tree
  - Locational Keying
  - Matsuyama's 2d-Tree
  - 4-D-Tree
  - R-Tree
  - R^+-Tree
  - Cell Tree
  - Spatial 2d-Tree/Trie

* After *B. C. Ooi*.

EXTENDED STRUCTURE ISSUES[*]

- efficient use of storage *vs.* ease of information retrieval
- techniques to extend point indexing
  - object mapping / transformation
  - object duplication / clipping
  - overlapping bounding rectangles
- organization
  - bucket
  - relational
  - grid

[*] After *B. C. Ooi*.

SPATIAL OPERATIONS

- *Item Deletion:* An item is removed from the data structure. This involves locating the item and then reorganizing the structure so that the item is no longer present. We anticipate that the deletion operation will not be of major importance for TRANSIMS applications.

- *Item Insertion:* An item is added to the data structure. This involves locating the place where the item will be added and then reorganizing the structure so that the item is there. We anticipate that the insertion operation will be used during the post-processing phase primarily for building indexes to simulation data output files; these are called “static” insertions.

- *Containment Search:* All items within a specified geographic area (usually rectangular or circular) are located. These searches will occur when it is necessary to find all of the vehicles, road segments, or other objects inside a specified part of a city.

- *Intersection Search:* All items intersecting a specified geographic object (usually a rectangle or circle) are located. Although, these searches are conceptually similar to containment searches, they generally must be implemented differently for most spatial data structures. It is not clear what role they will have in TRANSIMS.

- *Nearness Search:* The item nearest to a given point is located. The canonical example of this is locating the vehicle or road segment selected by the mouse in a GUI.

PERFORMANCE METRICS

- *Storage:* The number of bytes required to store/index a specified data set are counted. For most data structures, this can be done theoretically. This is mostly compiler independent (the only dependencies being the hidden bytes associated with virtual function tables and memory allocated using *new*). This is an important consideration for TRANSIMS because of the large amount of data to be indexed.

- *CPU Time:* The amount of processor time required to perform an operation outlined above is measured. This is both compiler and machine dependent, but we expect the relative performance of different data structures to be consistent. This is another important consideration for TRANSIMS and is in conflict with the requirement to minimize storage.

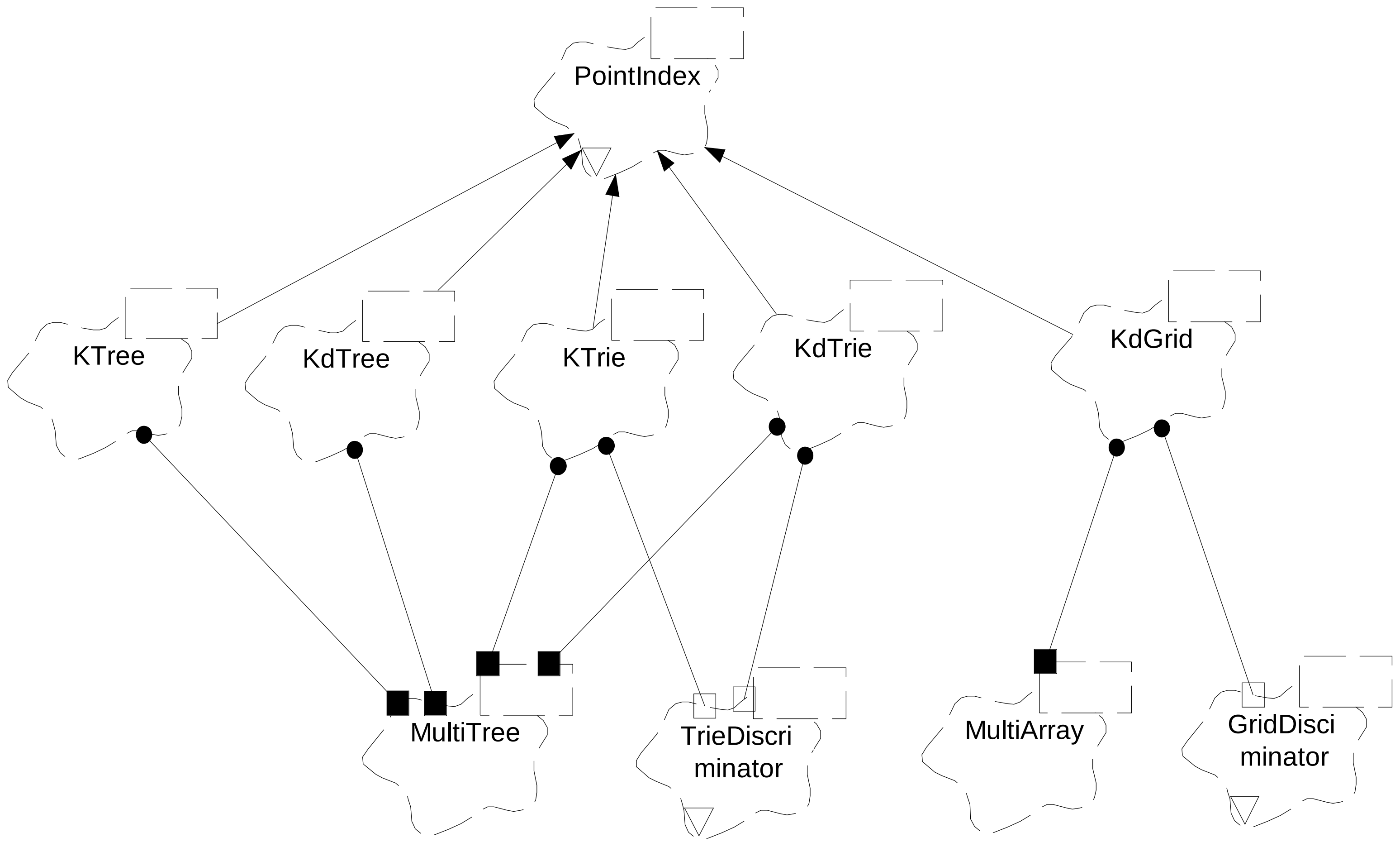
- *Wall-Clock Time:* The real time required to perform an operation outlined above is measured. This is highly machine, compiler, and operating-system dependent, but is of fundamental importance in an interactive computational environment.

- *Paging:* The number of times memory is paged during an operation outlined above is counted. This depends on the memory and disk architecture of the operating system. It may also be difficult to measure, but an understanding of the paging caused by various data structures and algorithms may be useful for optimization.

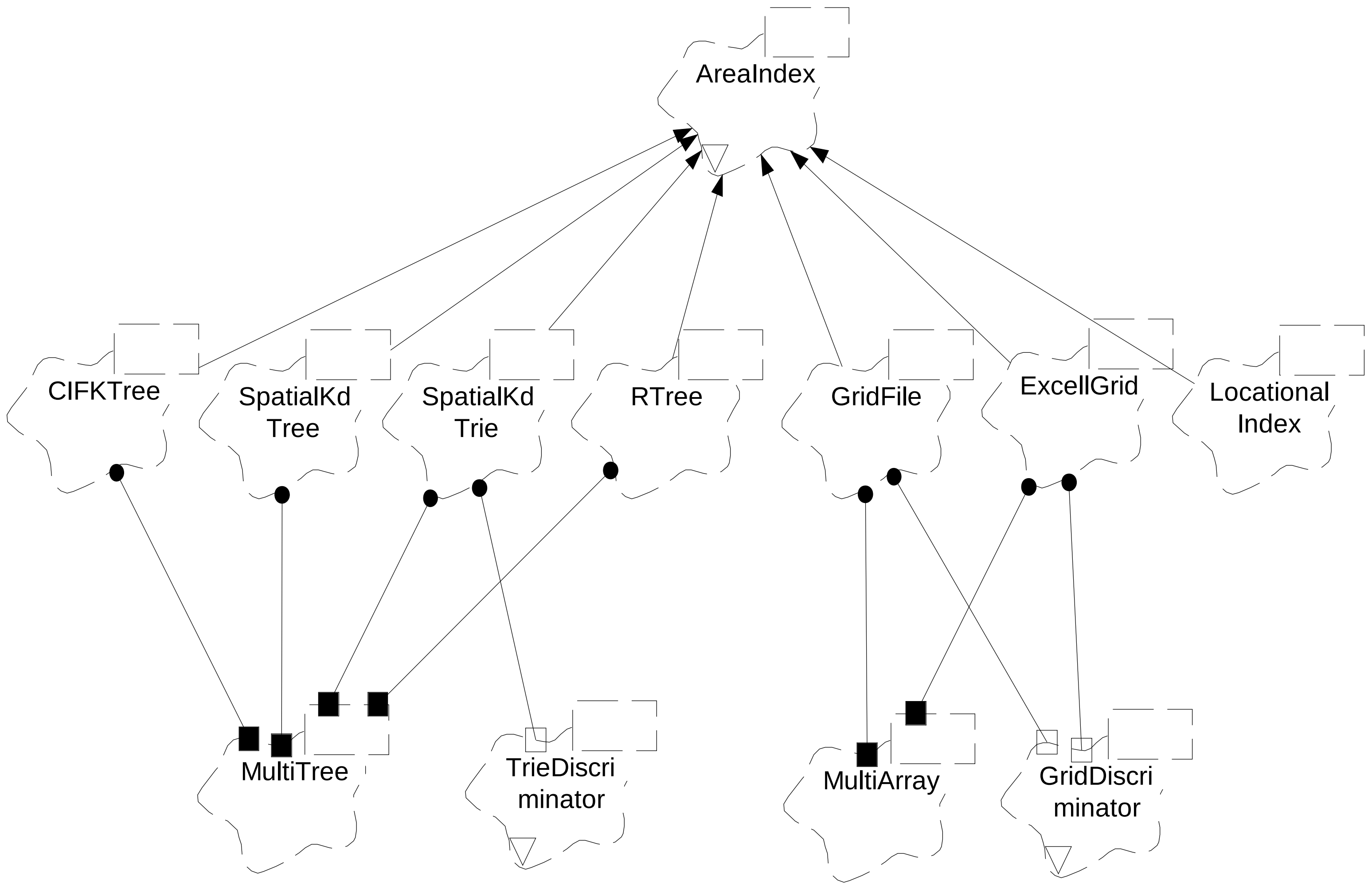
- *Order:* The number of nodes visited during an operation is counted. This can be done either theoretically or empirically.

- There will be other metrics relevant for distributed systems. We have not investigated these so far.

A DESIGN FOR POINT STRUCTURES

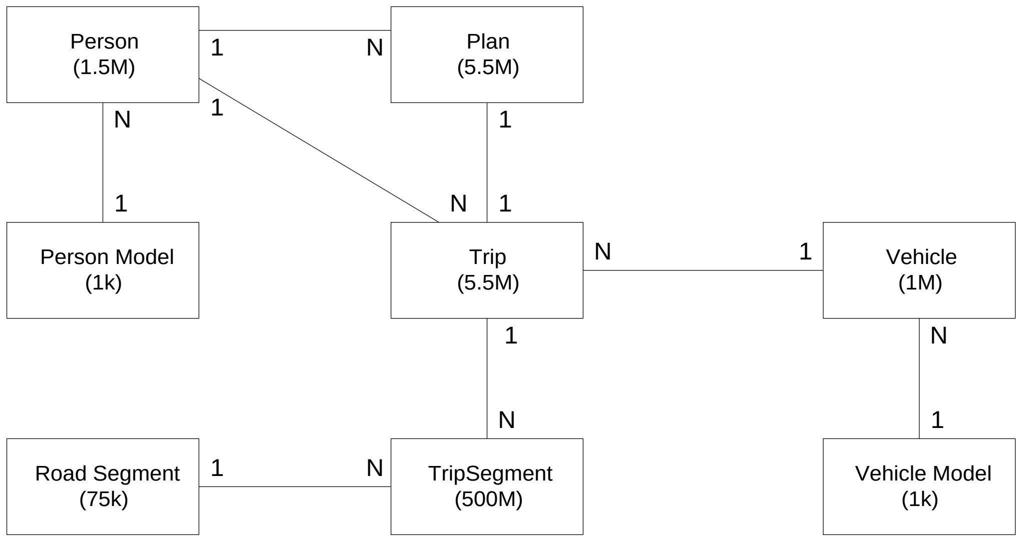


A DESIGN FOR EXTENDED STRUCTURES



DATABASE SIZE—ROUGH ESTIMATE

- sample ER diagram with number of entities for Portland, OR (24 hrs.)



- storing the trajectory-related (location, speed) data alone will require *at least* 25 GB if the data is packed, or 45 GB if it is not packed

CONCLUSION

- So far we have
  - defined issues related to spatial data structures
  - developed use cases
  - designed, implemented, and measured the performance of several candidate spatial data structures
  - designed and implemented utility classes needed for spatial data structures
  - studied the relationship of spatial data structures to nonspatial data structures and persistence mechanisms
- We next plan to
  - continue the design, implementation, and performance measurement
  - experiment with the use of spatial data structures in the context of different persistence mechanisms